Batch Script - Return Code

By default when a command line execution is completed it should either return zero when execution succeeds or non-zero when execution fails. When a batch script returns a non-zero value after the execution fails, the non-zero value will indicate what is the error number. We will then use the error number to determine what the error is about and resolve it accordingly.

Following are the common exit code and their description.

Error Code	Description
0	Program successfully completed.
1	Incorrect function. Indicates that Action has attempted to execute non-recognized command in Windows command prompt cmd.exe.
2	The system cannot find the file specified. Indicates that the file cannot be found in specified location.
3	The system cannot find the path specified. Indicates that the specified path cannot be found.
5	Access is denied. Indicates that user has no access right to specified resource.
9009 0x2331	Program is not recognized as an internal or external command, operable program or batch file. Indicates that command, application name or path has been misspelled when configuring the Action.
221225495 0xC0000017 -1073741801	Not enough virtual memory is available. It indicates that Windows has run out of memory.
3221225786 0xC000013A -1073741510	The application terminated as a result of a CTRL+C. Indicates that the application has been terminated either by the user's keyboard input CTRL+C or CTRL+Break or closing command prompt window.
3221225794 0xC0000142 -1073741502	The application failed to initialize properly. Indicates that the application has been launched on a Desktop to which the current user has no access rights. Another possible cause is that either gdi32.dll or user32.dll has failed to initialize.

Error Level

The environmental variable %ERRORLEVEL% contains the return code of the last executed program or script.

By default, the way to check for the ERRORLEVEL is via the following code.

Syntax

```
IF %ERRORLEVEL% NEQ 0 (
   DO_Something
)
```

It is common to use the command EXIT /B %ERRORLEVEL% at the end of the batch file to return the error codes from the batch file.

EXIT /B at the end of the batch file will stop execution of a batch file.

Use EXIT /B < exitcodes > at the end of the batch file to return custom return codes.

Environment variable %ERRORLEVEL% contains the latest errorlevel in the batch file, which is the latest error codes from the last command executed. In the batch file, it is always a good practice to use environment variables instead of constant values, since the same variable get expanded to different values on different computers.

Let's look at a quick example on how to check for error codes from a batch file.

Example

Let's assume we have a batch file called Find.cmd which has the following code. In the code, we have clearly mentioned that we if don't find the file called lists.txt then we should set the errorlevel to 7. Similarly, if we see that the variable userprofile is not defined then we should set the errorlevel code to 9.

```
if not exist c:\lists.txt exit 7
if not defined userprofile exit 9
exit 0
```

Let's assume we have another file called App.cmd that calls Find.cmd first. Now, if the Find.cmd returns an error wherein it sets the errorlevel to greater than 0 then it would exit the program. In the following batch file, after calling the Find.cnd find, it actually checks to see if the errorlevel is greater than 0.

```
Call Find.cmd

if errorlevel gtr 0 exit
echo "Successful completion"
```

Output

In the above program, we can have the following scenarios as the output −

- If the file c:\lists.txt does not exist, then nothing will be displayed in the console output.
- If the variable userprofile does not exist, then nothing will be displayed in the console output.

- If both of the above condition passes then the string “Successful completion” will be displayed in the command prompt.

Loops

In the decision making chapter, we have seen statements which have been executed one after the other in a sequential manner. Additionally, implementations can also be done in Batch Script to alter the flow of control in a program’s logic. They are then classified into flow of control statements.

S.No	Loops & Description
1	While Statement Implementation There is no direct while statement available in Batch Script but we can do an implementation of this loop very easily by using the if statement and labels.
2	For Statement - List Implementations The "FOR" construct offers looping capabilities for batch files. Following is the common construct of the ‘for’ statement for working with a list of values.
3	Looping through Ranges The ‘for’ statement also has the ability to move through a range of values. Following is the general form of the statement.
4	Classic for Loop Implementation Following is the classic ‘for’ statement which is available in most programming languages.

Looping through Command Line Arguments

The ‘for’ statement can also be used for checking command line arguments. The following example shows how the ‘for’ statement can be used to loop through the command line arguments.

Example

```
@ECHO OFF
:Loop

IF "%1"="" GOTO completed
FOR %%F IN (%1) DO echo %%F
SHIFT
```

```
GOTO Loop
:completed
```

Output

Let's assume that our above code is stored in a file called Test.bat. The above command will produce the following output if the batch file passes the command line arguments of 1,2 and 3 as Test.bat 1 2 3.

```
1
2
3
```

S.No	Loops & Description
1	Break Statement Implementation The break statement is used to alter the flow of control inside loops within any programming language. The break statement is normally used in looping constructs and is used to cause immediate termination of the innermost enclosing loop.